SFPtoPK

SFPtoPK converts HP LaserJet softfont files into TEX PK & TFM files

Version 2.0

Every reasonable effort has been made to assure the quality and completeness of this program, if you have any questions, comments, or suggestions or if you believe that you have found a bug please contact the author at the address given at the end of this document.

Changes (what's new and exciting)

Version 2.0

- Added support for HP AutoFont Support files.

Version 1.35

- Added mathmatical calculation of space parameters (see `/spacenorm` ...)
- Fixed bug that prevented `/spaceadj` from being negative
- Added space and space adjustment to configuration file

Version 1.34

- Made `SFPtoPK` report attempts to delete explicitly (re)mapped characters

Version 1.33

- Faster I/O on some softfonts, minor documentation changes

Version 1.32

- Included `PKscale` program for creating scaled `PK` files
- Improved speed of "scanning font" code for most fonts
- Fixed bug that caused `SFPtoPK` to miss some accent problems

Version 1.31

- Never officially released

Version 1.3

- First real release
- `SFPtoPK` didn't print a message if it couldn't find the kern file
- Added `none.lig`, `none.krn`, and `none.map`

Version 1.2

- Tentative real release, last beta

Versions prior to 1.2

- To save paper, changes made prior to version 1.2 are no longer reported

Usage

SFPtoPK is run from the DOS command prompt. In order to keep the program small and simple, a command-line interface has been chosen instead of something more user-friendly. Many of the options have reasonable default values and these can be set with a configuration file. The configuration file is described below. The general format for running SFPtoPK looks like this:

SFPTOPK softfont[.sfp] ⟨pkfile[.pk]⟩ ⟨plfile[.pl]⟩ ⟨/option1:value1⟩ ... ⟨/optionn:valuen⟩

Where:

softfont[.sfp] is the name of the HP LaserJet softfont file to convert to TEX format.

pkfile[.pk] is the name of the TEX PK file to create. By default, SFPtoPK will use the root name of the softfont file as the root name for the PK file.

plfile[.pl] is the name of the TEX PL file to create. In order to use the PK file created by SFPtoPK in TEX, you will also have to use the standard TEXware utility PLtoTF to create a TFM file from the PL file. By default, SFPtoPK will use the root name of the PK file as the root name for the PL file.

/option is one of the following options: /map, /lig, /minchar, /maxchar, /slantchar, /spacechar, /spaceadj, /spacenorm, /spacebold, /spaceital, /spacehmi, /xchar, /xadj, /krn, /krows, /kcols, /kpdots, /kmax, /kadj, /ktbl, or /auto. These options allow you to adjust the TEX font that SFPtoPK produces. Each of these options is described in more detail below.

Options

/map:filename

HP softfonts frequently contain many of the ligatures and accents TEX uses but they are in different ASCII positions; the /map option allows you to rearrange characters in the font. Each line in the map file should identify two characters (ASCII positions really), the character in the first position in the softfont will be moved to the second position in the PK file.

A word about characters: whenever SFPtoPK expects a character specification in a file (or a command line option), it can be done in four different ways. These four ways correspond to TEX notational conventions for indicating decimal, octal, and hexidecimal numbers and literal characters. The first way is to simply indicate the ASCII position with a decimal number (65, for example), the second way is to indicate the position with an octal number prefixed with a ' character ('101), the third way is to use an hexidecimal number prefixed with a " character ("41), and the fourth way is to use a character constant prefixed with a ` character (`A).

Consider, for example, the following lines from a map file:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This is a substitution map for \SFPtoPK.  This map is for HP softfonts
% with symbol set 10J (PS Text).
"A1 '074
170 '134
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

If this map file is used, the character at position A1 (hex) in the softfont file will appear in position 74 (octal) in the PK file (similarly for 170 (decimal) and 134 (octal), etc). As you can see, %-style comments are allowed in the map file.

The default mapping for all characters not present in the map file is to it's "natural" position, the same position it has in the softfont file, in the PK file. SFPtoPK will only use this mapping if the position is available in the PK file (*i.e.* if no other character has been mapped to that position).

If the second "character" in the map-pair is the word REMOVE, the softfont character will not be present in it's natural position in the PK file even if the position is available.

Mapping has a higher precedence than removing. An attempt to remove a character that has been explicitly mapped will produce a warning message and the character will not be removed. For example, if the map file indicates that the character at position "7F should come from position "BA in the softfont and that the character at position "7F should be removed, the character will come from "BA even if the remove command appears later in the map file. It is obviously silly to attempt to do both, and the warning message will give you a clue that you have done something wrong. (I spent more than an hour looking for a bug in SFPtoPK before I discovered that I'd made a typo in my map file ...).

The map file also allows several options for manipulating character accents. When the character is mapped to it's new position, you may specify that only the accent or only the character is to be moved. It is even possible to specify that only baseline accents (like the cydil "¸") should be moved. In this way it is possible to form dotless ı's and ȷ's for TEX and to take accents that are not present as individual symbols off of characters that have them. In order for SFPtoPK to steal an accent (or remove it), there must be at least one row of blank dots between the accent and the character in the character bitmap.

The map options for accent manipulation are ACC, NOACC, and BLA. These options should be placed after the second character in the map-pair.

Here are some examples of how the map options can be used. The REMOVE option is used to make sure that the close curly brace and the underscore character do not appear in the TEX font (presumably they are being removed because the raised dot and backwards quote accents were not available to replace them), the cydil, open circle, and "check" accents are being taken off of characters in the softfont and the dotless i and j are being formed from the regular "i" and "j" in the font.

```
"7D   Remove       % I don't want ``}'' in the font
'137  Remove       % I don't want ``_'' in the font either
"B5   '030  bla    % BaseLineAccent cydil
"D4   '027  acc    % ACCent, raised o accent
"EC   '024  acc    % ACCent, upside down ^
`j    '021  noacc  % NO ACCent, dotless j
`i    '020  noacc  % NO ACCent, dotless i
```

By default, SFPtoPK will look for a map file whose name is the same as the symbol set of the HP softfont that it is converting (0U, 1U, 0A, 8M, etc). For example, if SFPtoPK is converting a softfont with the symbol set 10J and you do not specify a /map option when you run SFPtoPK, it will look for 10J.MAP. If you do not give a path, SFPtoPK looks for map files in the current directory then the subdirectory specified by the "SFPTOPKPATH" environment variable. Under DOS 3.0 or higher, SFPtoPK will look in the same directory as its EXE file if the SFPTOPKPATH environment variable is not specified. Under DOS 2.xx, SFPtoPK will only look in the current directory if SFPTOPKPATH is undefined.

/lig:filename

The /lig option allows you to specify TEX ligature commands for the PK file that you are creating. The format of the ligature file is similar to the format of the map file. On each line in the file, three characters must be specified; a ligature will be created for the first two characters to produce the third. For example, the following ligature file:

```
% sample lig file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
‘f    ‘i    ’014            % fi
‘f    ‘l    ’015            % fl
```

specifies the “fi” and “fl” ligatures.

SFPtoPK uses the same search strategy for finding the ligature file as it does for finding the map file, with the exception that the default extension for ligature files is .LIG.

/minchar:character

The /minchar option allows you to limit the range of characters created in the PK file. No character whose position in the PK file falls below /minchar will be included. For example, in a standard TEX font, if /minchar:’040, the PK file will not include most of the accents or ligatures or the Greek letters.

/maxchar:character

The /maxchar option is analogous to the /minchar option. No character whose position in the PK file falls above /maxchar will be included. In general, /maxchar is more commonly used than /minchar. For example, if you want to exclude the “upper” 128 characters of a softfont from your PK file (except the ones you’ve mapped into standard TEX positions), you can set /maxchar:127 to keep them out.

/slantchar:character

The /slantchar option defines the character that SFPtoPK will use to determine the slant of the font. By default, the “[” character (or the character specified by the configuration file) is used. SFPtoPK computes the slant by finding the upper-left most dot and the lower-left most dot in the character bitmap, for this reason, the character selected should not have any curves or serifs on it’s leading edge. In general, a tall character is better than a short one. The open square bracket and the vertical bar (if present in the softfont) are both good choices, curly braces, parenthesis, slashes, and rounded or serifed alphabetical characters are bad choices.

HP AutoFont Support may override this option.

/spacechar:character

The /spacechar option (and the /spaceadj option) allow you to tweak the size of an interword space in TEX for the font that you are creating. In older softfonts, the default value (which is the default HMI or "pitch" of the font) is frequently inaccurate (this is not so much the case with newer fonts).

The /spacechar option lets you specify which character in the font should be used for the width of an interword space. For example, `/spacechar:‘i` specifies a lower case I and `/spacechar:32` specifies the ASCII space character in the softfont. Note: unless you specify the space character, it will *not* be used, even if it exists in the font. By default, `SFPtoPK` always uses the default HMI specified in the softfont header.

HP AutoFont Support may override this option.

/spacenorm, /spaceital, /spacebold

First, a little background. The interword spacing in TEX is very dynamic. Versions of `SFPtoPK` prior to 1.35 used either the default HMI of the softfont or some other (explicitly requested) character width to calculate the value of an interword space. The values of the stretch and shrink components of the TEX interword space were always constant fractions of the space value. I've reached the conclusion that these methods for calculating the TEX space parameters are unsuccessful. Far too often, `PK` fonts created using the default HMI as the space factor have interword spaces that are much too small.

What are the other alternatives? I did a little investigation into the space parameters used in the CM family of fonts. I discovered that the space parameters fit very well onto a smooth curve. The precise curves vary for each of the parameters and for each of the three "kinds" of fonts (normal, bold, and italic) available in the CM family but they all fit very well.

The /spacenorm, /spacebold, and /spaceital options are used to tell `SFPtoPK` to calculate the space parameters (space, shrink, stretch, extra space, and quad) with these formulas instead of using any specific character width. Since the CM family of fonts does not include any bold-italic fonts, a seperate formula is not provided for that kind of font.

How well will this work? Only time will tell. I suspect that it will provide much better space parameters for most fonts. The equations used to calculate the space values are well defined over the range of TEX fonts that I have (roughly 5pt to 17pt) and I expect them to perform well outside that range but it is possible that they will not work well at extremely large sizes. However, for large display-sized faces it is probably less important to have finely tuned space parameters anyway.

The /spaceadj option can still be used to tweak individual fonts. It may turn out, in practice, that all fonts need to be hand adjusted to create the best possible appearance. Hopefully these refinements to `SFPtoPK` have improved the computed values to the point where it is not intolerable to use them without adjustment.

HP AutoFont Support may override this option.

/spacehmi

The /spacehmi option is provided so that you can override the default method (in the configuration file) for calculating TEX space widths. If the "space" configuration variable is not used, /spacehmi is the default method.

HP AutoFont Support may override this option.

/spaceadj:number

The /spaceadj option lets you make a second adjustment to the width of an interword space. The /spaceadj parameter can specify either a number of PCL dots of space or a percentage of the current interword space. If it is followed by a `%` character, it is a percentage. If the number is preceded by a plus or minus sign, the parameter is relative to the current size, otherwise it represents an absolute size. A few examples will illustrate: `/spaceadj:+50%` makes the width of a space 50% larger than its current value, `/space:20%` makes the width of a space one fifth of it current value, `/space:+5` makes it five *dots* wider, `/space:5` makes it *five dots wide* (very narrow indeed). This adjustment is applied after the space has been calculated using the method specified by other options (if any).

/xchar:character

The /xchar option (and the /xadj option) let you adjust the X-height. By default, `SFPtoPK` uses the height of the character in the ASCII position of a lower case "x" ('116) for the X-height. If the font does not contain a lower case "x", then the x-height from the softfont header is used. The /xchar option can be used to specify which character to use instead of an "x".

HP AutoFont Support may override this option.

/xadj:number

The /xadj option lets you make adjustments to the X-height. Like the /spaceadj option, it can be either relative or absolute and may specify a number of PCL dots or a percentage of the current value.

Note: the X-height parameter is a bit counter-intuitive. If you make the X-height larger, you effectively move accent marks *down*, not up as you might expect. The X-height specifies the height of characters for which TEX *does not* need to raise the accent marks.

Here are a few examples: `/xadj:+10%` raises the X-height 10 percent (effectively lowering accent marks), `/xadj:-4`, makes the X-height 4 pixels shorter (effectively raising accent marks).

HP AutoFont Support may override this option.

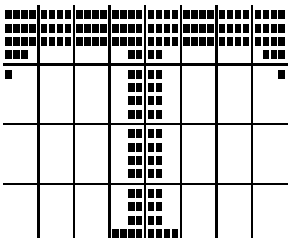
/auto:filename

The /auto option specifies that HP AutoFont Support files should be used. The filename specified on the /auto option should be the name of the GLUE.TXT file in your AutoFont directory. If you store your AutoFont metrics files in the directory `C:\AUTOFONT`, you should specify `/auto:c:\autofont\glue.txt` to turn on HP AutoFont support. The GLUE.TXT file should have been created by the program that installed your AutoFont Support files. Using GLUE.TXT is is the standard way of finding AutoFont Support files for each font.

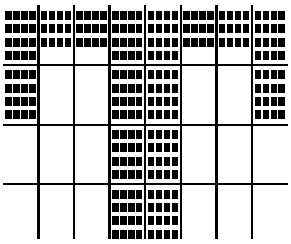
Kerning

Automatic support for kerning is still under development. That is to say, I still don't know how to make it "do what you mean" whenever it kerns letter-pairs. Described below is the algorithm that `SFPtoPK` uses. If you can think of any improvements to this algorithm, or if you notice any distinct patterns in the way that the various parameters interact, please drop me a line and I'll be happy to improve it.

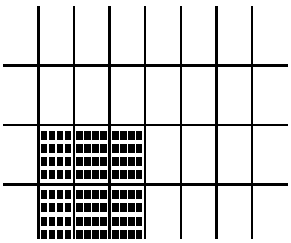
`SFPtoPK` begins by placing a grid over each letter. Each square in the grid is either available or used, depending on whether there are *any* black pixels in that square. Consider the letter T:



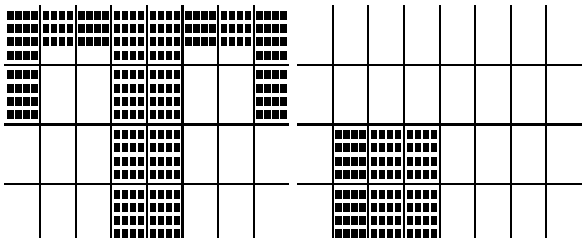
For the purposes of this algorithm, the letter T has this shape:



Now, consider a lowercase letter. For the purpose of this example, a lowercase letter (*e.g.* "a" or "o") has the following shape:



Finally, consider these two letters next to each other.



The first rough estimate of "kernability" is calculated using the number of blank grid squares between each character. In the example above, the "T" has (0,0,3,3) blank grid squares on its right hand side (counting

from top to bottom) and the lowercase "o" has (8,8,1,1) blank grid squares on its left hand side. The row with the smallest sum provides the initial kern estimate (the sums above are (8,8,4,4), making 4 the rough estimate). If the smallest sum is zero, the letter-pair will not be kerned.

The rough estimate can be refined in two independent ways: it is possible to reinsert space between two kerned letters, and it is possible to limit the maximum amount of kerning.

Kerning in `SFPtoPK` is controlled by a kern file (that describes which letters should be kerned) and five parameters. The parameters are:

/KROWS	The number of rows in the grid
/KCOLS	The number of columns in the grid
/KMAX	The maximum number of PCL *dots* of kerning allowed
/KADJ	The number of PCL *dots* to insert back between kerned letters
/KPDOTS	The number of *plus* dots of kerning to insert (described below)

In addition to kerning letters closer together, TEX provides for the possibility of kerning letters farther apart. `SFPtoPK` supports this kind of kerning in a manual fashion. If you place a plus sign (`+`) after the letter-pair in the kern file, `SFPtoPK` will insert /kpdots of extra space between the two letters. In order to provide some flexibility, `SFPtoPK` allows you to place an arbitrary number of plus signs after the letters in the file. `SFPtoPK` will place /kpdots of extra space between the letters for each plus sign.

The format of the kern file is similar to the format of the map and ligature files. Consider the following fragment of a .KRN file:

```
‘f‘)++
‘A‘t
‘a‘j+
```

Each line contains a pair of characters (expressed in any of the accepted forms of notation). If the letter-pair is followed by one or more plus signs, the characters will be kerned apart, otherwise they will be kerned together. The three lines shown above indicate that "f)" will be kerned apart by /kpdots*2 dots, "At" will be kerned together (according to the algorithm discussed above) and "aj" will be kerned apart /kpdots dots.

Just to keep the record straight, the inspiration for this algorithm came from the AGFA Compugraphic Division's document describing the IntelliFont Scalable Typeface Format.

The only remaining kerning parameter is /ktbl. This option lets you specify the filename for a "kerning information" output file. If the filename is non-blank, `SFPtoPK` will create a TEX file that illustrates all of the kerns and ligatures that it created in the `PK` file as well as showing the sample text from the standard "testfont.tex" file.

HP AutoFont Support

The addition of HP AutoFont support is a large step forward in the successful translation of LaserJet softfonts to professional-quality typesetting fonts. One of the primary disadvantages of LaserJet softfonts is that they contain relatively little information about the metrics of the font. Font metrics form the basis of accurate typesetting. In fact, as TEX makes clear, the typesetter doesn't need to know anything about the shape of the characters in each font, it only needs the metrics! TEX uses only the `TFM` files when constructing the DVI file. It is the DVI-to-printer program that needs PK files.

The lack of metrics in a softfont file is the primary reason why `SFPtoPK` has so many options. Each one is an attempt to provide a missing font metric.

HP created AutoFont Support files in an effort to provide font metrics for all applications that use softfonts. `SFPtoPK` can use these metrics to construct TEX `TFM` files (well, `PL` files really, but...).

A Word of Warning:

HP AutoFont Support files have the extension `.TFM`, however, they are not the same as TEX `TFM` files! Do not confuse HP AutoFont metrics files with TEX `TFM` files. In this document, HP AutoFont metrics files will never be referred to as TFM files!

How `SFPtoPK` finds AutoFont Support files:

HP AutoFont Support files are quite complex. Many different fonts can use the same support file. For this reason, a special text file is created by the program that creates AutoFont Support files. This file is called `GLUE.TXT` and it is stored in the same directory as your AutoFont Support files. This file contains an entry for each softfont that points to the appropriate AutoFont Support file. `SFPtoPK` reads this file to determine which AutoFont Support file to use. If the font you are converting does not have an entry in the `GLUE.TXT` file, `SFPtoPK` ignores the AutoFont option.

`SFPtoPK` reads the HP AutoFont Support file and uses the metric values in that file to construct the TEX `PL` file. Every metric specified in the AutoFont Support file overrides any other setting of that metric by `SFPtoPK`. It is far more likely that the AutoFont Support file is correct. Some metrics are required in an AutoFont Support file, however, not all metrics are in all files (many are optional). If a particular metric is not present, `SFPtoPK` will use the value that it calculated (based upon options, etc.) as if AutoFont Support was not being used.

AutoFont Support files are discussed in more detail in the documentation for `PKtoSFP`. `PKtoSFP` has the much more difficult task of constructing and installing valid AutoFont Support files.

Configuration File

To make SFPtoPK easier to use, default parameters can be stored in a configuration file. The configuration file is a simple DOS text file. Each line in the file defines exactly one parameter. Every parameter has a program name, a parameter name and a value. SFPtoPK searches for parameters by program name and parameter name. The program name is optional, if it is not specified the parameter matches all program names. This is a very general format with a full potential that is not realized by SFPtoPK. This format is designed so that different programs can share the same configuration file and some or all of the same configuration parameters.

In cases where an option can be specified in *both* the configuration file *and* as a command line option, the command line option will take precedence.

If SFPtoPK is run under DOS 3.00 or later, it looks for the configuration file in the same directory as the SFPTOPK.EXE file. Otherwise, SFPtoPK looks in the current directory. In either case, you can tell SFPtoPK explicitly what configuration file to use by setting the DOS environment variable "SFPTOPK" equal to the fully qualified name of the configuration file.

Configuration Parameters

SFPTOPK MINCHAR=character

Defines the smallest character to be placed in the PK file.

SFPTOPK MAXCHAR=character

Defines the largest character to be placed in the PK file.

SFPTOPK SLANTCHAR=character

Defines the default /slantchar.

SFPTOPK SPACE=space-option

Defines the default method for calculating interword spacing. The legal values are: norm, bold, and ital which select the mathmatical methods explicitly, math which selects the "appropriate" mathmatical method, hmi which selects the softfont default HMI value, or any valid TEX character specification. The math option will use the italic formula if the font is italic, the bold formula if the font is bold (or bold-italic), and the normal formula if it is neither.

SFPTOPK SPACEADJ=adjustment-value

Defines the default space adjustment value. This option has exactly the same values as the /spaceadj parameter discussed above.

`SFPTOPK SFDIR=`directory

Defines the directory where softfonts are kept. `SFPtoPK` will look there if it cannot find the softfont in the current directory.

`SFPTOPK AUTOFONT=`d:\path\glue.txt

Defines the default setting of the AutoFont Support feature. If you wish to enable AutoFont support, the filename specified should be the pathname of your AutoFont `GLUE.TXT` file. Refer to the discussion of the `/auto` option for more information.

If you specify *any other* filename, `SFPtoPK` will use the name you specify as the name of the HP AutoFont support file. This would be a very foolish thing to do in the configuration file!

`SFPTOPK KRN=`filename

Defines the name of the default kern file. I created the default kern file by reverse engineering the CMR10 font. For the English language, I expect that it is complete but you will almost certainly want to create other kern files for different languages.

`SFPTOPK KROWS=`number

Defines the number of rows in the grid that is laid over each character. This is one measure of the roughness of the grid.

`SFPTOPK KCOLS=`number

Defines the number of columns in the grid that is laid over each character. Together with `KROWS`, this determines the roughness of the initial kern estimate. The finer the grid, the closer together the initial estimate is likely to kern.

`SFPTOPK KPDOTS=`number

Defines the number of dots to insert between characters that are kerned apart. As described in the section on Kerning, `SFPtoPK` will insert /kpdots of blank space between two characters for each plus sign that follows the character-pair in a krn file.

`SFPTOPK KMAX=`number

Defines the maximum number of dots to kern. Setting this parameter prevents some kerning combinations (like "P.") from becoming *far* too close together.

`SFPTOPK KADJ=`number

Defines the number of dots to reinsert between letters after they have been kerned together. This will never move them farther apart than their natural distance. (that is, if they are only going to be kerned three dots closer together and KADJ is four, they *will not* be moved one dot farther apart).

`SFPTOPK KTBL=`filename

Defines the default output filename for the "kern information" file.

A Real Life Example

Here is a brief description of how I use `SFPtoPK` and `MergeSFP` (another program available from Small Planet Software) to create a TEX softfont from a group of HP softfonts. In my case, I have HP's TypeDirector program for creating typefaces from scalable font data. One advantage of this program is that it allows me to create fonts from several different symbol sets. In this way, I can collect a group of softfonts from the same typeface that contain all of the necessary characters for TEX. It has been pointed out, since I wrote this, that TypeDirector can be configured to produce a single font that has all of the necessary TEX characters, but the general principle still holds.

Here is the group that I use: symbol set 0U (US ASCII) for the basic alphabet, symbol set 6M (Ventura Math) for the greek letters, symbol set 6J (Microsoft publishing) for the "f" ligatures, and symbol set 10J (PS Text) for the accents and several other ligatures.

First, I create (or collect) all of these fonts at the size I want to use in TEX. If I create 10pt CG Times fonts in these symbol sets with TypeDirector, I will get four fonts: `trr14usa.sfp`, `trr14vma.sfp`, `trr14mpa.sfp`, and `trr14tsa.sfp`.

Next, I run `MergeSFP` as so:

```
MERGESFP 0x temp_0x trr14usa trr14vma trr14mpa trr14tsa
```

This creates an output softfont called `TEMP_0X.SFP` that contains all of the characters that I need for TEX. Next, I run `SFPtoPK`to create the TEX `PK` and `PL` files.

```
SFPTOPK temp_0X cgtms10
```

`SFPtoPK`uses the `0X.MAP`, `0X.LIG`, and `DEFAULT.KRN` files to place the characters in the appropriate places in the `PK` file and to create the appropriate kerns and ligatures. The only character that is missing is the cross-bar used in "Ł" and "ł" in TEX (I can't find anything appropriate in a standard font).

I use symbol set 0X for a very specific reason. `SFPtoPK` will look for map and ligature files with a root name of 0X when I run it because that is the symbol set of this font. However, the symbol set 0X is not strictly legal (to the HP LaserJet printers) so I know that I will never have a collision between my munged-together font and some real softfont.

The map, lig, krn, and mrg files that I use to do this are included in `SFPtoPK` as an example.

Bye Bye

I hope that you find `SFPtoPK` useful. `SFPtoPK` is absolutely free. You may copy it and give it away to anyone that you think might benefit from it. However, you *may not* sell it or profit from it's distribution in any way, shape, or form.

If you wish to contact the author, you may write to:

Norman Walsh
#42I Southwood Apartments
Brittany Manor Drive
Amherst, MA 01002
USA

or send electronic mail to:

walsh@cs.umass.edu

This electronic mail address requires access to Internet domains (through BITNET or UUCP, for example). This is possible from CompuServe and from several large national BBS systems as well as many colleges and universities.

Other Programs by Small Planet Software

`PKtoSFP`

`PKtoSFP` converts TEX `PK` fonts into HP LaserJet softfonts. `PKtoSFP` can produce HP AutoFont Support files to provide accurate spacing and kerning information to other applications. `PKtoSFP` is the inverse of `SFPtoPK`.

`PKscale`

Changes the magnification of TEX `PK` files. This can be especially useful if you are working with devices of differing resolutions. `PKscale` is included in the `SFPtoPK` package.

`MergeSFP`

Merges multiple LaserJet softfonts into a single file. If you are generating TEX fonts, you may discover that you need characters from several different symbol sets (and, hence, several different LaserJet softfonts) in order to create a complete TEX character set. `MergeSFP` allows you to construct a single LaserJet softfont containing the appropriate characters from several different softfonts.

Sfware

The Sfware utilities allow you to download, rotate, compress, expand, view, and perform special effects on softfonts. The effects provided include bold, fill, convert to fixed spacing, halftone, hollow, invert, mirror, outline, convert to proportional spacing, resize, reverse, shade, shadow, slant, stripe, tilt, three-d, hollow-three-d, and filled-three-d effects. The effects can be tailored and customized for any font with various parameters and shading patterns. Sfware is distributed under a shareware license agreement.

SFP2Auto

SFP2Auto reads HP LaserJet softfonts and produces HP AutoFont Support files directly. Many applications that cannot use softfonts directly, can install them with HP AutoFont Support. For example, this program allows you to install arbitrary softfonts into WordPerfect using only the PTR program!